Interakce člověk–počítač v přirozeném jazyce (ICP)

LS 2013 — Dekódování

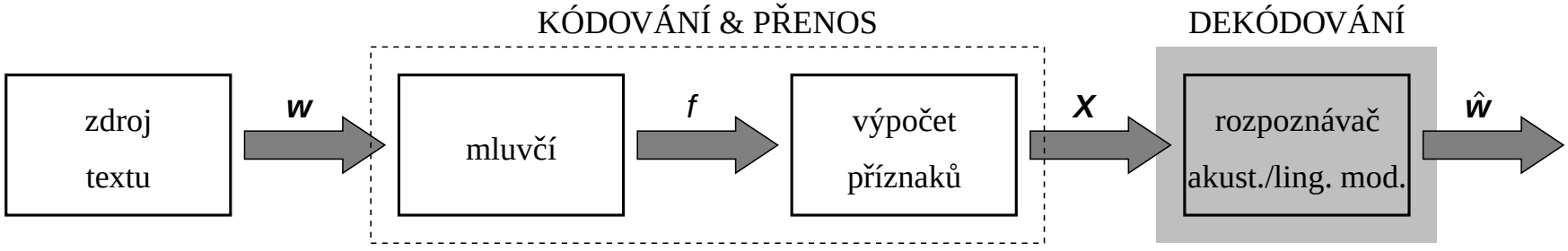
Tino Haderlein, Elmar Nöth

Katedra informatiky a výpočetní techniky (KIV)
Západočeská univerzita v Plzni

Lehrstuhl für Mustererkennung (LME)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Dekódování

- Dekódování je vlastní proces rozpoznávání řeči, při kterém hledáme mluvenou posloupnost slov pomocí akustického a lingvistického modelu.
- Výraz pochází z teorie přenosu zpráv.
- Optimálně: dekódování pomocí Bayesova pravidla
- Problém: V kontinuálním rozpoznáváním řeči musíme vyšetřovat všechny kombinatoricky možné posloupnosti slov a délky slov.



Dekódování

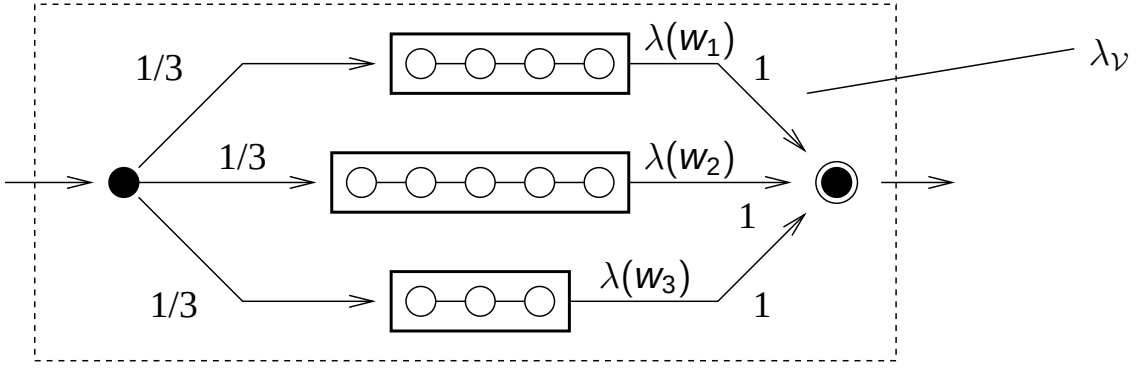
- Dokonce aplikace dynamického programování (Viterbiův algoritmus) pro hledání je při dlouhých nahrávkách a mnohých slovech ještě příliš náročná, co se týče paměti a doby výpočtu.
- Řešení: kombinace různých metod redukce složitosti
  - účinné algoritmy vyhledávání, např. dynamické programování (DP) a $A^\star$
  - pruning: nezohledňuje určité alternativy podle heuristických kritérií
  - postupné rozetření: nejprve užíváme např. bigramové modely, pak přesně hledáme pomocí trigramů
  - zmenšení vyhledávací oblasti: vhodná kombinace různých alternativ, např. slov se stejnými hláskami na začátku slova
  - dobrá implementace
- Problém: Kvůli suboptimálnímu dekódování teď nejsou jen chyby modelací, ale i dekódováním.

Synchronní vyhledávání

- Pro synchronní vyhledávání svazujeme HMM slov a tím vytvoříme **kompilovanou mříž**.
- Tato mříž je sama HMM, jen že ji nikdy nevyrobíme úplně, ale jen částečně podle toho, která slova a které přechody mezi slovy právě zpracováváme.
- **Synchronní** vyhledávání: Vyhledáváme zleva doprava ve směru času.
- Běžně jsou to rozšiřování a varianty Viterbiova algoritmu.

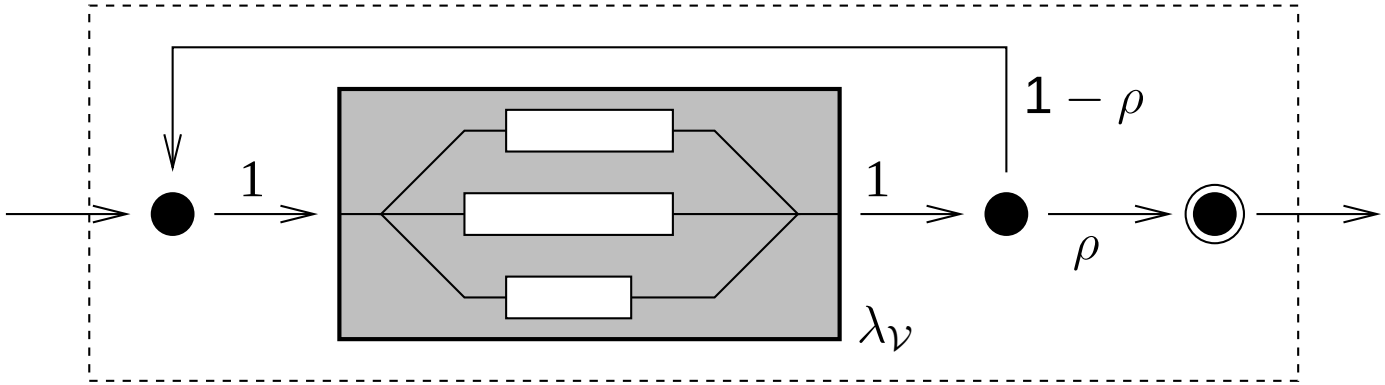
Mříže HMM

- Modely $\lambda(w_i)$ slov pro všechna slova rozpoznávacího slovníku ($w_i \in \mathcal{V}$) svazujeme a tím vzniká mříž $\lambda_{\mathcal{V}}$.
- Pro rozpoznávání užíváme např. Viterbiův algoritmus:
  Pro pozorování hledáme nejpravděpodobnější posloupnost slovních modelů v mřížkovém HMM $\lambda_{\mathcal{V}}$.
- HMM $\lambda(w_i)$ svazujeme **konfluentními stavy**, tj. stavy, které nemají výstupní rozdělení.
- Algoritmus vyhlédávání nesmí přiřazovat konfluentní stavy příznakovým vektorům.



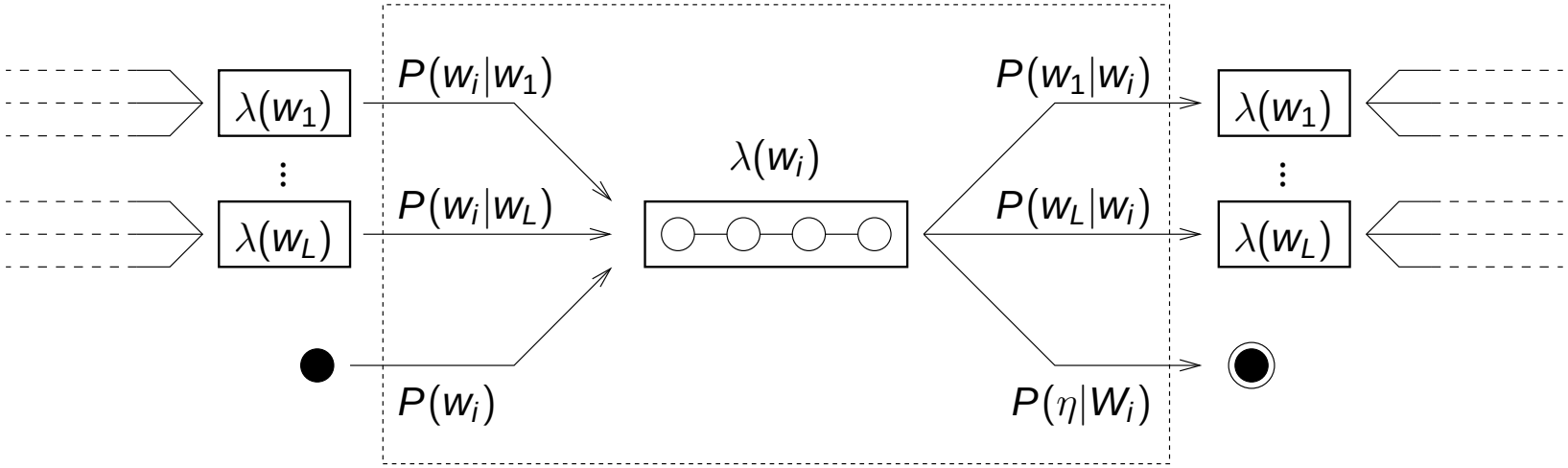
Mříže HMM: věty s libovolnou délkou

Konfluentní stavy sníží komplexitu mříže, protože stavy na koncích slov jsou sdružovány:



Mříže HMM: věty s libovolnou délkou

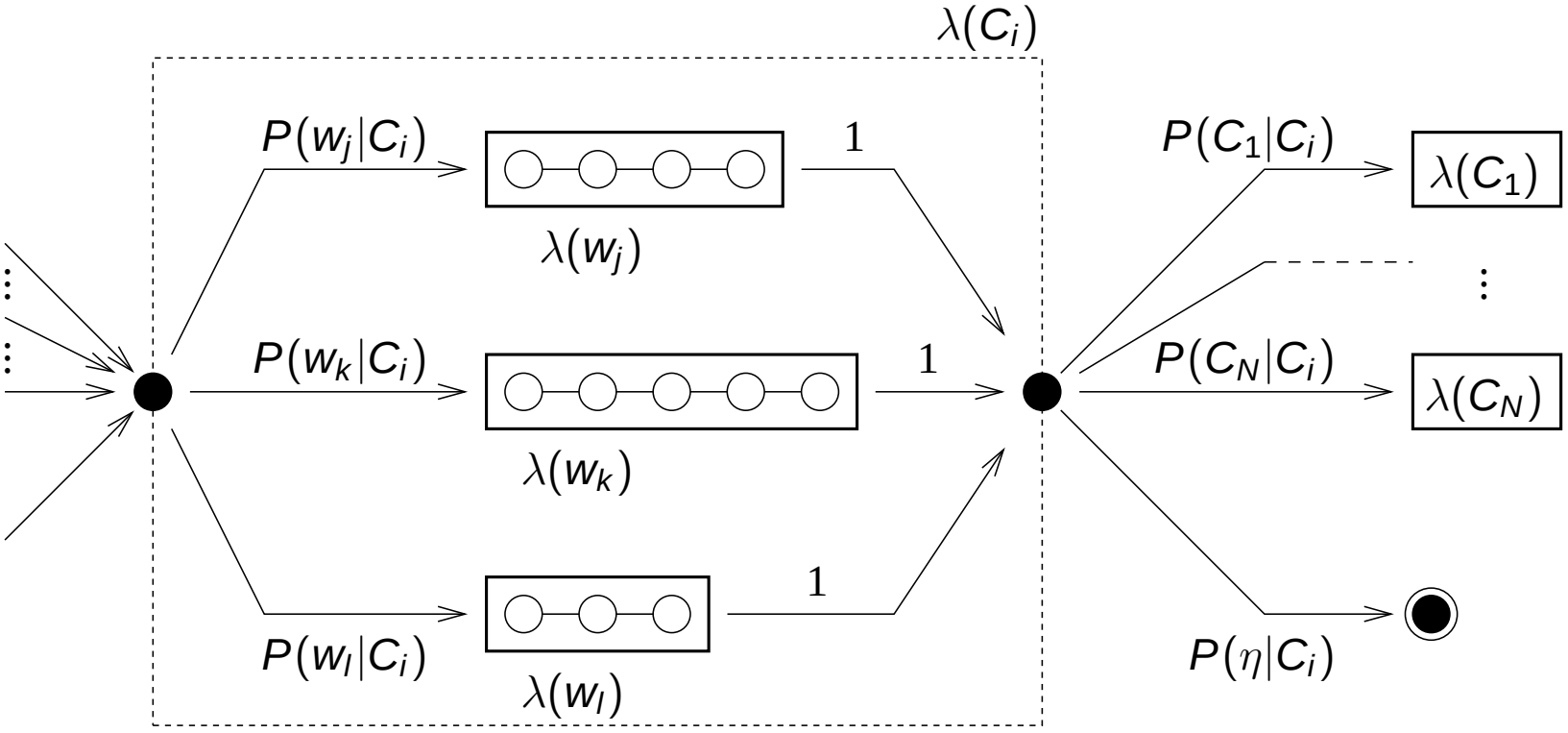
Při použití bigramové mříže musíme vyrábět všechny přechody slovo-slovo:



Slova jedné kategorie můžeme ale sdružovat v podmřížích
$\rightarrow$ redukce komplexity.

Mříže HMM: kategoriální bigramy

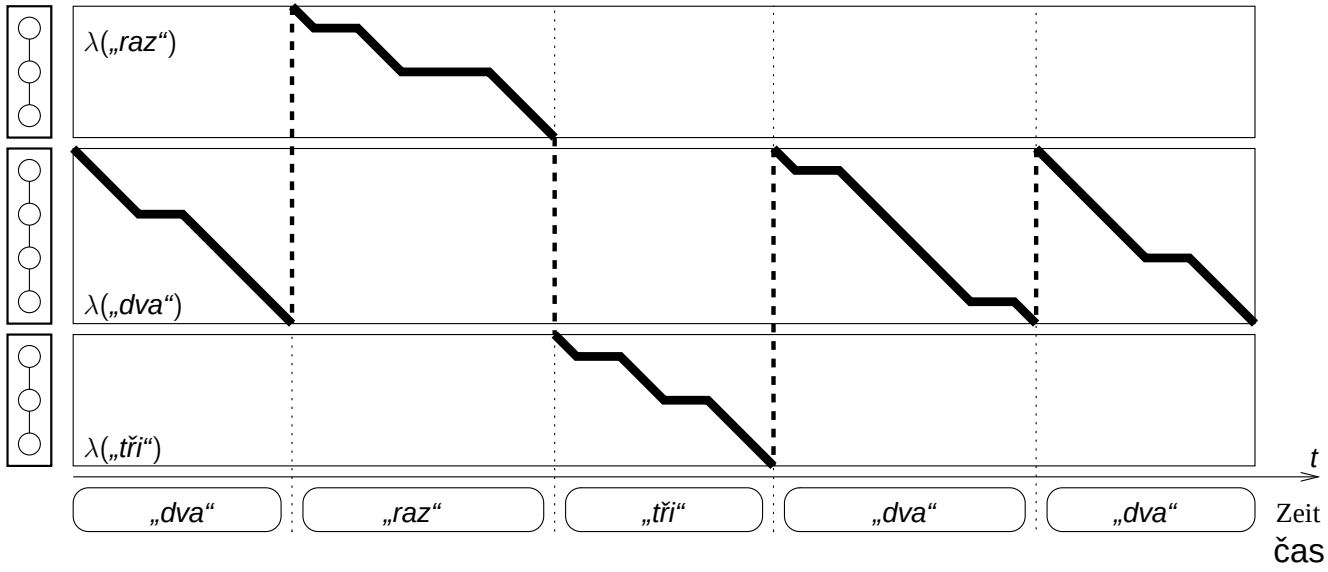
podmříže pro každou kategorii



Dobově synchronní vyhledávání: Viterbiův algoritmus

- nejjednodušší metoda dobově synchronního dekódování
- stanov nejpravděpodobnější posloupnost stavů $\boldsymbol{q}^*$ pro pozorování $\boldsymbol{X}$
- výsledek rozpoznávání: posloupnost slov $\boldsymbol{w}^* = w(\boldsymbol{q}^*)$

optimální posloupnost stavů pro promluvu: skoky ne uvnitř slov, ale na úzlech na konci slova k následujícím libovolným slovům



Dobově synchronní vyhledávání: dekódování dopředu

- rozšíření Viterbiova algoritmu: uvnitř slova nehledáme nejlepší posloupnost stavů, ale počítáme dopřednou pravděpodobnost
- tj. maximalizace jen u slovních přechodů; sumace uvnitř slova

Dobově synchronní vyhledávání: dekódování dopředu

1. inicializace: $\vartheta_1(j) = \pi_j b_j(\boldsymbol{x}_1)$, $\psi_1(j) = 0$ pro všechna $j = 1, \ldots, N$
2. iterace pro $t = 0, ..., T-1$; pro všechny stavy $j = 1, \ldots, N$ stanov

$$\psi_t(j) = \operatorname*{argmax}_i \vartheta_{t-1}(i) a_{ij}$$

$$\vartheta_t(j) = \begin{cases} \max\limits_i & (\vartheta_{t-1}(i) a_{ij}) \cdot b_j(\boldsymbol{x}_t), \quad \text{když } s_j \text{ je stav na začátku slova} \\ \sum\limits_i & (\vartheta_{t-1}(i) a_{ij}) \cdot b_j(\boldsymbol{x}_t) \quad \text{pro všechna ostatní } s_j \end{cases}$$

3. terminace: stanov $P^*(\boldsymbol{X} \mid \boldsymbol{\lambda}) = \vartheta_T(N)$ a $q_T^* = \psi_T(N)$
4. zpětné následování optimální posloupnosti slov:
   pro $t = t-1, \ldots, 1$ stanov $q_t^* = \psi_{t+1}(q_{t+1}^*)$
5. řešení posloupnosti slov: $\boldsymbol{w}^* = w(\boldsymbol{q}^*)$

Dobově synchronní vyhledávání: Beam Search

- Problém: Při Viterbiově nebo dopředném hledáním musíme zohlednit příliš hodně cest $\rightarrow$ pomalé.
- Pro $t = 0, ..., T - 1$ a všechny stavy j

$$\vartheta_t(j) = \max_i(\vartheta_{t-1}(i)a_{ij}) \cdot b_j(\boldsymbol{x}_t)$$

- Řešení: Heuristickou metodou rušíme „beznadějné“ alternativy co nejdříve, jen nadějné alternativy zůstanou v „paprsku“ (beam) $\mathcal{O}_{t-1}$ vyhledávání.

$$\vartheta_t(j) = \max_{i \in \mathcal{O}_{t-1}} (\vartheta_{t-1}(i)a_{ij}) \cdot b_j(\boldsymbol{x}_t)$$

- Přitom se ovšem může stát, že eliminujeme cestu, která má na začátku špatné hodnocení, později je ale optimální $\rightarrow$ ztratíme nejlepší řešení $\rightarrow$ chyby rozpoznávání.

Dobově synchronní vyhledávání: Beam Search

- Problém: Čím víc alternativ vyšetřujeme, tím menší je pravděpodobnost, že mylně eliminujeme optimální cestu, ale tím pomalejší je vyhledávání.
- Potřebujeme kompromis mezi přesností a účinností.
- Špatné kompromisy jsou konstantní prahy vzhledem k množině aktivních stavů nebo k minimálnímu hodnocení stavů.

Dobově synchronní vyhledávání: Beam Search

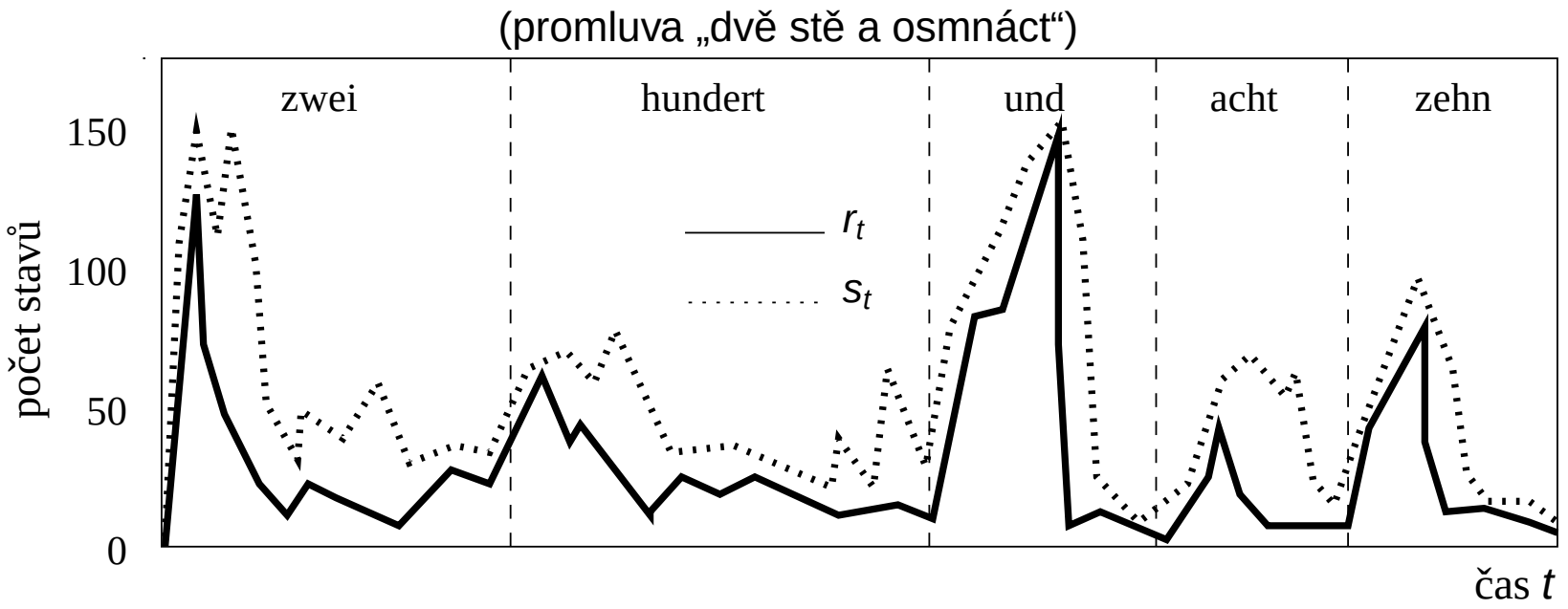
- Beam Search: Časově závislý práh pravděpodobnosti vzhledem k hodnocení aktuálně nejlepší cesty:

$$\mathcal{O}_t = \{i | \vartheta_t(i) \geq B_0 \cdot \Lambda_t\}, \quad \text{kde} \quad \Lambda_t = \max_k \vartheta_t(k)$$

- B_0 je velmi malé kladné číslo (závislé na aplikaci!), např. $B_0 = 10^{-22}$; čím menší je B_0, tím pomalejší (a lepší) je výsledný rozpoznávač.
- Když je hodnocení nejlepšího stavu dobré, beam se automaticky zúží (obsahuje méně stavů).
- Když je hodnocení nejlepšího stavu špatné, vyhledávání přijímá víc stavů.

Beam Search: pracovní postup

- typicky: šířka paprsku se silně obměňuje a je typicky velmi velká na začátcích slov
- charakteristické veličiny:
  - **rozsah** r_t stavu q_t^* globální nejlepší posloupnosti
  - **šířka paprsku** s_t: počet zbývajících výpočtů hodnoty $\vartheta_t(j)$
- ideální: šířka s_t blízko u grafu rozsahu, ale není menší



Beam Search: účinná realizace

- problém účinnosti: určení maxima možné až tehdy, až všechna hodnocení známa
- řešení: užívání parciálního maxima $\tilde{\Lambda}_t \leq \Lambda_t$, které je během výpočtu hodnocení pořád znovu stanovováno
  $\rightarrow$ beam je nepatrně širší než je to nutné

Beam Search: účinná realizace

1. inicializace: stanov $\mathcal{O}_0 = \{1\}$ a pro všechna t: $\tilde{\Lambda}_t = 0$
2. iterace: pro všechna $t = 0, \ldots, T-1$ a pro všechna $i \in \mathcal{O}_t$ (při seřazení seznamu):

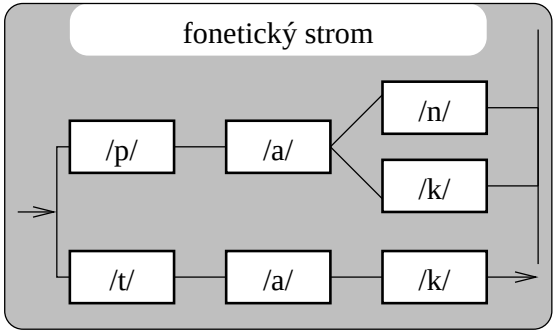
   ($\star$) pro všechna j s $a_{ij} > 0$:
   - $\vartheta = \vartheta_t(i) \cdot a_{ij} \cdot b_j(\boldsymbol{x}_{t+1})$
   - když $\vartheta < B_0 \cdot \tilde{\Lambda}_{t+1} \rightarrow$ pokračuj u ($\star$)
   - když $j \in \mathcal{O}_{t+1}$ a $\vartheta \leq \vartheta_{t+1}(j) \rightarrow$ pokračuj u ($\star$)
   - $\vartheta_{t+1}(j) = \vartheta$ a $\psi_{t+1}(j) = i$
   - když $j \notin \mathcal{O}_{t+1}$, pak přidej j na hlavu/začátek (když $\vartheta > \tilde{\Lambda}_{t+1}$) popř. na konci seznamu $\mathcal{O}_{t+1}$
3. terminace a zpětné řetězení/následování: jako při Viterbiově algoritmu

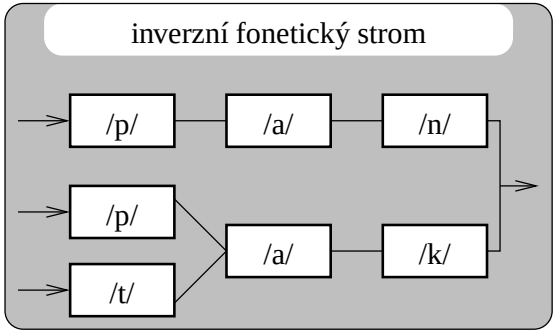
Organizace slovní zásoby

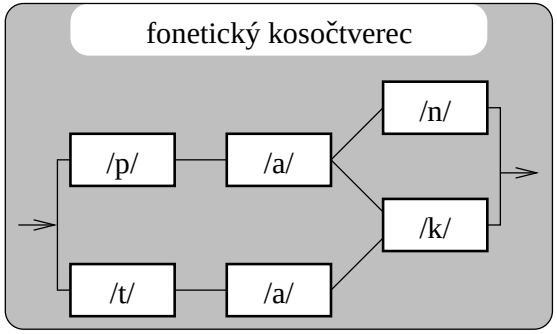
- Cíl: zrychlení vyhledávání zabráněním zbytečným vícenásobným výpočtům
- K tomu snížíme počet stavů v kompilované mříži HMM.
- Pozorování: **Ekvivalence prefixů**, tj. hodně slov má stejný začátek, např. „pan“ a „pak“.
- Když jsou začátky slov v mříži HMM reprezentovány jen společnými uzly, pak musíme hodnotit HMM pro hlásky /pa/ jen jednou.
- **Fonetický strom**: Každá cesta odpovídá fonetickému základnímu tvaru slova; na listech stojí akustické hodnocení slov.
- Můžeme strom generalizovat i pro kontextově závislé modely.

Fonetický strom

struktury kompilované mříže HMM:

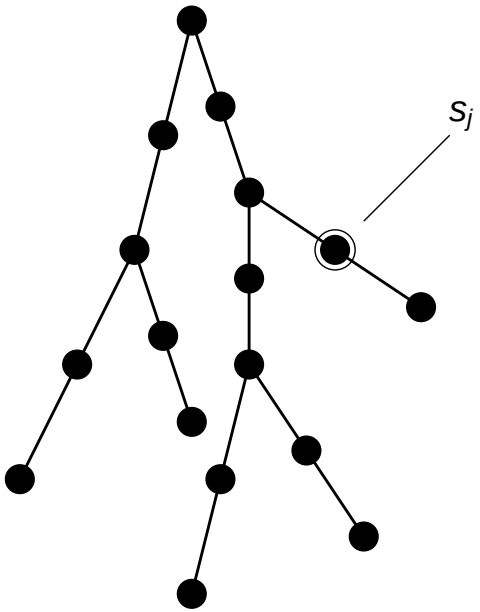


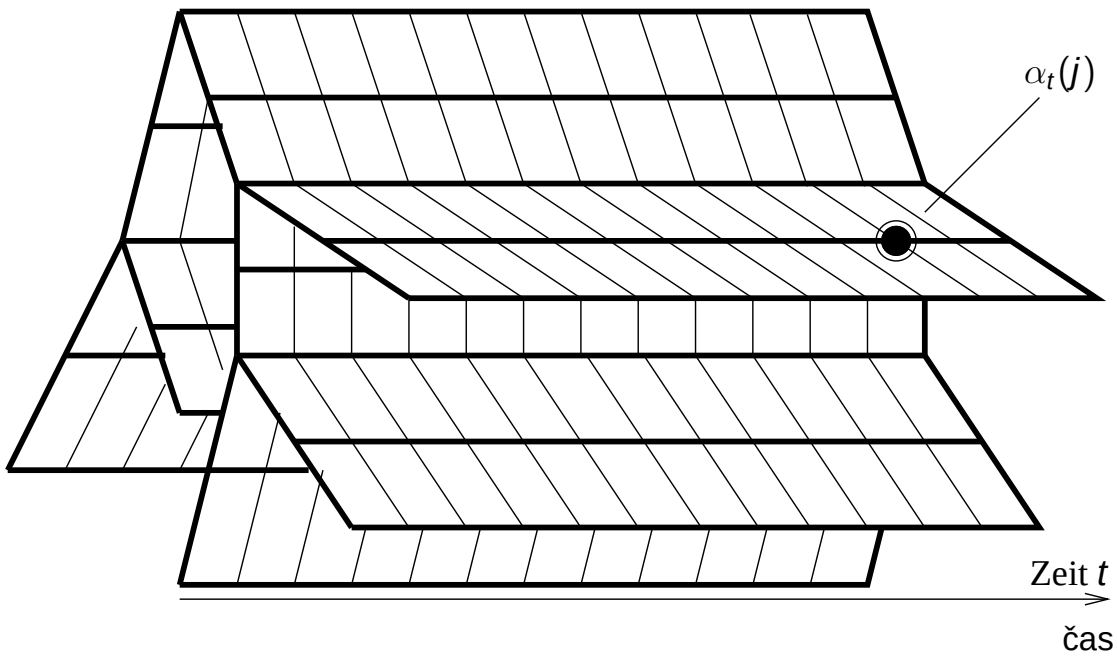




Fonetický strom: rozpoznávání jednotlivých slov

- Uzly stromu jsou expandovány do hláskových modelů.
- Matice dopředných nebo Viterbiových hodnocení má tvar jako na obrázku dole.
- Redukce oblasti vyhledávání je výrazná, když je slovní zásoba velká.





Fonetický strom: rozpoznávání kontinuální řeči

- Problém: Hodnocení na uzlech a listech fonetického stromu obsahují i hodnocení jazykovým modelem.
- Hodnocení jazykovým modelem je závislé na předchozím slově.
- Pro každé předchozí slovo musíme vytvořit kopii fonetického stromu $\rightarrow$ náročné.
- Při bigramu tím potřebujeme $|\mathcal{V}|$ kopií.
- Počet kopií se sníží, jakmile jsou cesty eliminované z „paprsku“.
- Při užívání určitých heuristik stačí průměrně pouze dvě instance fonetického stromu v jedné časové jednotce.

Kombinace $P(\boldsymbol{X} \mid \boldsymbol{W})$ a $P(\boldsymbol{W})$

- Teoreticky optimální podle Bayesova pravidla: kombinace pravděpodobností akustického a jazykového modelu násobením
- Praxe ale ukažuje, že se odhadnuté hodnoty $P(\boldsymbol{X} \mid \boldsymbol{W})$ a $P(\boldsymbol{W})$ hrubě odlišují od skutečnosti:
  - N-gramy zohledňují jen část kontextu.
  - Příznakové vektory x_t a x_{t-1} jsou závislé.
- Odchylky působí poškození výsledku rozpoznávání.
- Heuristické řešení: slovní pokuta (insertion penalty) a váha jazykového modelu (language [model] weight, linguistic matching factor, linguistic weight)

Insertion penalty

- Jazykový model $P(\boldsymbol{W})$ má také funkci pokutovat vkládání nových slov.
- Když je např. jazykový model uniformní, tj. $P(w_i) = \frac{1}{L}$, pak prostřednictvím jazykového modelu dostane pouze cesta, do které je vloženo nové slovo, trochu horší hodnocení než jiná cesta bez tohoto nového slova.
- Když je pokuta za vkládání nového slova příliš vysoká, bude rozpoznávač generovat spíše méně ale zato delší slova.
- Když je pokuta malá, bude rozpoznávač dávat přednost mnoha krátkým slovům.
- Parametr slovní pokuta $0 < \rho \leq 1$ je váha, se kterou násobíme hodnocení jazykového modelu; tak je možné mezi oběma pády najít rovnováhu.
- $\hat{P}(w_i|w_{i-n+1}, ..w_{i-1}) = P(w_i|w_{i-n+1}, ..w_{i-1}) \cdot \rho$
- $\hat{P}(w_1, w_2, ...w_N) = P(w_1, ...w_N) \cdot \rho^N$

Insertion penalty: příklad

„**parketovat**“ mnohými krátkými, akusticky hodícími se slovy:

Uhr in ja am ja am den morgen um am ja ein Zug nach ja am es Uhr
[*hodin v ano ve_dne ano ve_dne toho zítra v ve_dne ano jeden vlak do ano ve_dne ono hodin*]
($\rho = 1$)

Uhr dem fahren wann wie morgen vormittag ein Zug nach am Uhr
[*hodin tomu jet kdy jak zítra dopoledne jeden vlak do ve_dne hodin*]
($\rho = 10^{-2}$)

Uhr den fahren wann geht morgen vormittag ein Zug nach Frankfurt
[*hodin toho jet kdy půjde zítra dopoledne jeden vlak do Frankfurt(u)*]
($\rho = 10^{-4}$)

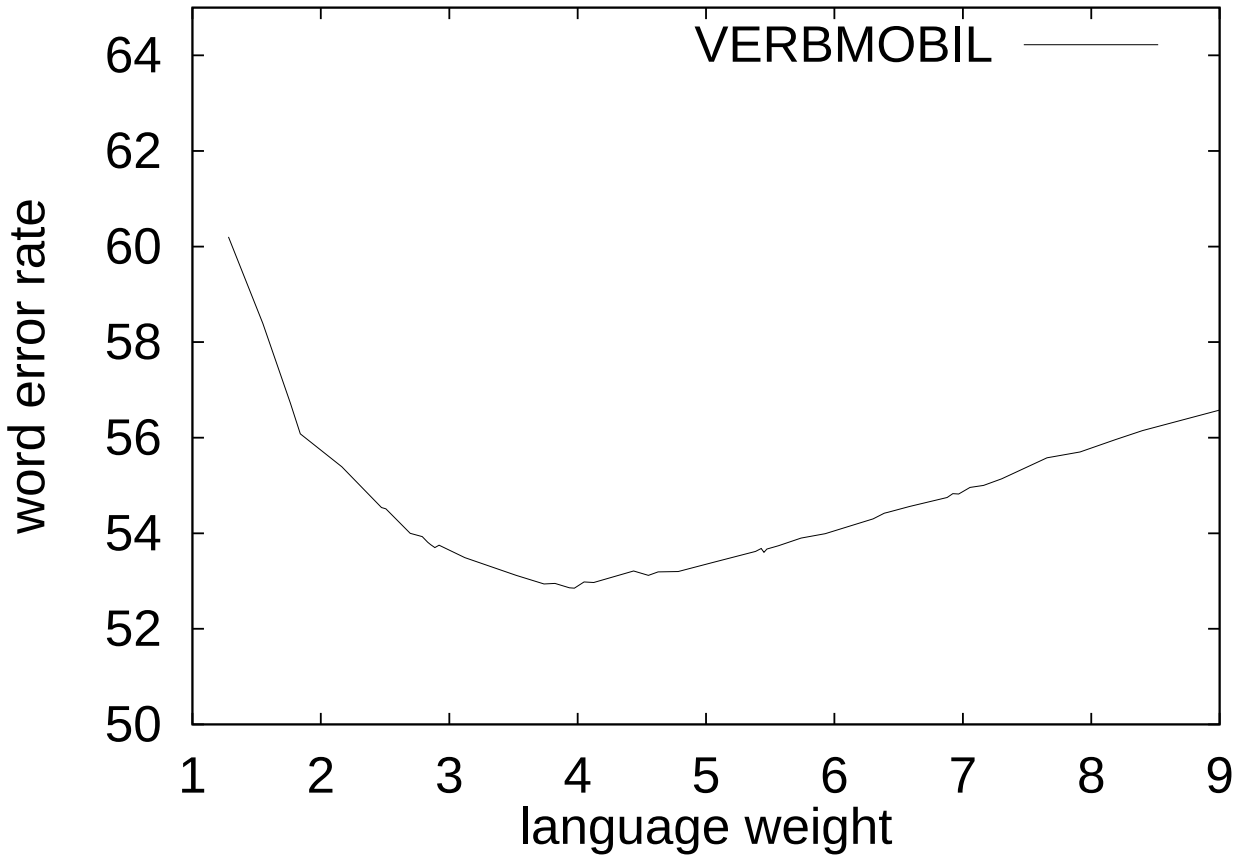
guten Tag wann geht morgen vormittag ein Zug nach Frankfurt
[*dobrý den kdy půjde zítra dopoledne jeden vlak do Frankfurt(u)*]
($\rho = 10^{-6}$)

Language Model Weight

- HMM podceňuje akustické hodnoty hustot/pravděpodobností, tj. typicky je akustické hodnocení výrazně menší než hodnocení jazykovým modelem.
- Kromě toho jsou variace, tj. rozsahy hodnot hodnocení akustického a jazykového modelu, různé.
- Vyrovnání hodnot je možné pomocí language (model) weight LW.
- $\hat{P}(w_i|w_{i-n+1},..w_{i-1}) = P(w_i|w_{i-n+1},..w_{i-1})^{LW} \cdot \rho$
- Hodnota LW je závislá na aplikaci a typicky $LW > 1$, např. 4 nebo 7.
- $P(w_i|w_{i-n+1},..w_{i-1}) << 1$, proto sníží $LW > 1$ hodnocení jazykového modelu a adaptuje ho tím na akustické hodnocení.
- Mezi ρ a LW existují ovšem vzájemná ovlivnění, obě spolu musíme optimalizovat na validačním vzorku.

Language Model Weight: příklad

Evaluace výkonu rozpoznávače v závislosti na language weight;
velká slovní chybovost (word error rate, WER) → špatný rozpoznávač:



Postupné rozetření: vícefázové dekódování

- v literatuře také nazývano jako two-pass nebo multi-pass decoding
- Komplexita algoritmu beam search silně stoupá při komplexnějších informačních zdrojích (trigramech, 4-gramech atd.).
- Řešení: Hrubý průběh rozpoznávání s bigramy generuje množinu slovních a větných hypotéz.
- Nejjednodušší možností jsou přitom tzv. n-best-lists, tj. seřazené seznamy těch N posloupností slov, které mají nejvyšší hodnocení.
- Do slovní mříže (word lattice) jsou různé slovní hypotézy doplněny se svými počátečními a konečnými dobami.
- Slovní graf zobrazuje slovní hypotézy jako uzle v grafu.
- Generování slovní mříže nebo n-best-list např. rozšířením Viterbiova algoritmu: Místo nejlepšího předchůdce uložíme *k* nejlepších předchůdců.

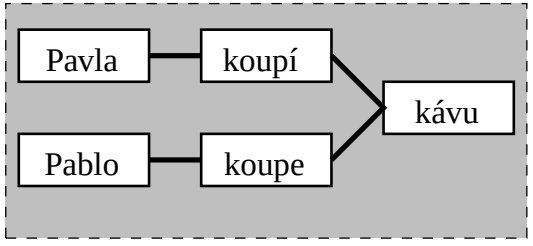
Postupné rozetření: vícefázové dekódování

Příklady různých možností, jak reprezentovat výsledky prvního průběhu rozpoznávače:

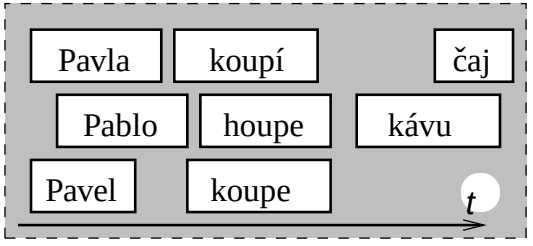
POSLOUPNOSTI SLOV

1. Pavla koupí kávu
2. Pavla koupe kávu
3. Pablo koupe kávu
4. Pavla houpe kávu
5. Pavel koupí kávu

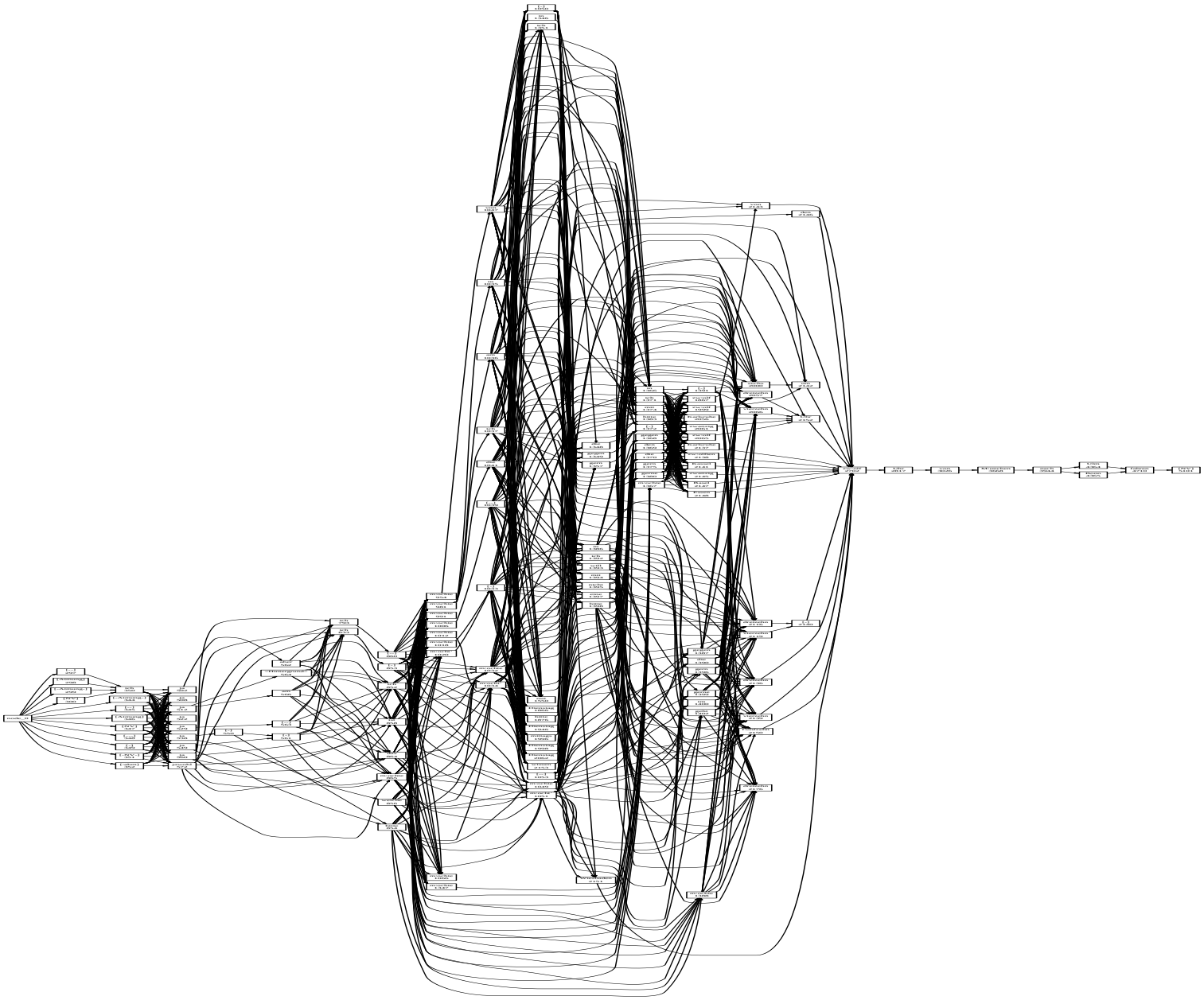
SLOVNÍ GRAF



SLOVNÍ MŘÍŽ



Slovní graf



Postupné rozetření: vícefázové dekódování

- V druhém průběhu rozpoznávače pak znovu hodnotíme posloupnosti vět v n-best-list pomocí tri- nebo 4-gramu; příklad: rozpoznávač Byblos (kolem 1992):

  Beam search užívá akustický model s monofony, stromovou strukturu a bigramový jazykový model; seznam nejpravděpodobnějších posloupností slov je znovu zhodnotěn pomocí trifonových akustických modelů, hranic slov a trigramového jazykového modelu.
- Alternativa: Slovní graf prohledáváme algoritmem A* (asynchronní vyhledávání).
- Složitosti postupu vypočítáme akustickými hodnoceními a trigramovými jazykovými modely.
- Zbytkové náklady (costs) můžeme odhadovat hodnoceními bigramu.
- Tento odhad ale obecně **není** optimistický!